a dispatcher 605 serving the computer to see if attention is required by any of the devices 190 residing lower in the hierarchy. If top byte 612 is found to be set, then the next lower level or middle tier 611 is interrogated. The middle tier 611 includes vectors of multiple local summary bytes 615a-615n. Finally, the bottom tier 610 includes completion vectors 618a-618n which contain one byte 620a-620n per device. Devices 190 set these detailed completion vector bytes 620a-620n in the bottom tier 610 to inform the processor 130 of I/O completion events. There is one local summary byte 615a-615n for each completion vector 618a-618n respectively, with each completion vector 615a-615n representing multiple devices 190. The number of devices within a completion vector is processor dependent (for instance, based upon cache line size). In one embodiment, optimized processor dependent instructions are used to perform the scanning of the completion vector bytes.

[0034] Each device 190 is assigned a unique completion vector byte 620a-620n, its associated local summary byte 615a-615n, and the single global summary byte 612. The device 190 is totally unaware that the completion vector byte may be in close proximity with completion vector bytes assigned to other devices. This invention assumes that separate per-device "queues" 220 and 222 are used between the processor 130 and the I/O device 190 to contain the detailed status describing which of the pending I/O events have completed. This invention only deals with the initiative passing piece of I/O completion. The preferred implementation of this invention assumes that devices can atomically update host memory at a byte level of granularity, but the concepts would equally apply to any higher / lower level of atomicity.

[0035] To close serialization timing windows, the three levels 610, 611 and 612 of completion bytes must be set by the device 190 in a well defined order. Specifically, device 190 must first set its respective completion vector byte 620, followed by the completion vector's respective local summary byte 615, and finally the global summary byte 612. The processor 130 must reset these bytes in the reverse order. This may result in an over initiative condition (i.e. the processor thinks there is new work to be performed, when it has already processed that work during a previous completion vector scan).

[0036] Significant cache line contention on the global / local summary bytes (updated by the devices) can be avoided by having the devices first read the bytes before attempting to store into them. The update is only performed if the target byte is not already set. This will cause the summary bytes to effectively become read only, from the time that they are set by any one device to the time that they are reset as part of the dispatcher poll processing. The timing windows described above are all satisfied as long as the reads are implemented in the order described (low tier 610 to high 612).

[0037] Referring to Fig. 10, a process is established at 650 to determine which device or devices 190 need to be serviced by the dispatcher 605. At 652, the buffers are appended to the detailed I/O queue 220 or 222, as part of the send/receive operation. At 654, the device's completion vector 620a is set, whereupon it's summary byte 615a is set at 656, and the global byte 612 is set at 658, in low to high order. At 660, the dispatcher 605 polls the global summary byte 612 and finds it is set. At 662, the dispatcher 605 then interrogates the respective summary bytes 615a-615n, and finally at 664 interrogates the respective completion vectors 618a-618n, and their bytes 620a-620n to service the device 190, and resets the bytes in high to low order. The reset instructions must not complete until the updated bytes are made visible to the other processors in the system (i.e. out of L1 cache) in order to insure I/O impetus is never lost.

[0038] Since each device is assigned a unique lowest level completion vector byte, the control information describing the device can be easily obtained by maintaining a parallel vector of control block addresses. Specifically, once it is seen that completion vector byte 44 (for example) is set, that byte offset can be used as an index into an array of entries (each 4 or 8 bytes in length depending upon the addressing range), that contains the address of a control block that describes the device now known to be requiring attention.

[0039] The three tiered hierarchy of I/O completion vectors scales horizontally in addition to vertically. Specifically, the horizontal scaling allows a single hypervisor that supports multiple concurrent guest OS images within it (e.g. IBM's z/VM), to efficiently determine which guest requires dispatching to process the pending I/O initiative in the completion vectors. In this environment, the global summary byte is owned by the hypervisor, and each middle tier summary byte is owned by a unique guest OS. The hypervisor dispatcher then uses the middle tier summary byte to determine which guest the pending I/O initiative is targeted for. The devices storing into the I/O completion vectors are ignorant of this additional dispatching level.

[0040] Fig. 11 illustrates the host computer 210 having an OS which includes the processor 700 which executes dispatcher software 605 (see Fig. 10). As explained in connection with Fig. 10, the host computer 210 includes a heirarchy 600 whose highest level includes a global summary byte (GSB) 612. As explained, whenever a device 190 requires attention, the bytes in the hierarchy 600 are set from low order to high, until the GSB 612 is set. The host computer 210 also includes a Time-of-Day (TOD) register 670 in which is recorded the last time the GSB 612 was set, and a Target-Delay-Interval (TDI) register 672 for storing a target-delay-interval value specified by the OS. These two values are shared across all devices implementing the low level interrupt. In one preferred implementation, to minimize cache line accesses, these two registers 670 and 672 reside in the same cache line 674 as the GSB 612 itself. This allows for read-before-write activity for two purposes: first, to avoid heavy write access to that cache line 674; and second, to obtain both the last time the GSB 612 was set (possibly by another device 190), and the delay value in TDI register 672 that is to be enforced. Only the device 190 that sets the GSB 612 is responsible for storing the time-of-day value in the TOD register 670. All others should just perform the comparison with the current TOD, to determine if an interrupt is required.

[0041] Also included is a clock 678 for containing the current time-of-day value. When the operating system is initialized, the present time-of-day value is placed in the TOD register 670 as shown at 680, and a time delay interval value is placed in the TDI register 672. The devices 190 are then associated with individual vectors 620a-620n as represented by 675 and previously explained. As represented by 677, as part of completing send/receive I/O operations, each device reads the global byte cache line 674. If the GSB 612 is set, the device subtracts the last time-of-day value in the TOD register 670 from the current time-of-day value, and, if the result is greater than the target-delay-interval value in the TDI register 672, a low level interrupt is sent to the I/O processor 700 of the host computer 210 by hardware of a device adapter 191 which connects the device 190 to the computer 210, as represented by 679. It will be understood that the device adapter 191 may be a separate apparatus, or could be built into the device 190, as may